COMP 576 - Fall 2017
Assignment 1

Alberto Fung - NetID: af31

October 5, 2017

Backpropagation in a Simple Neural Network

1a) Dataset

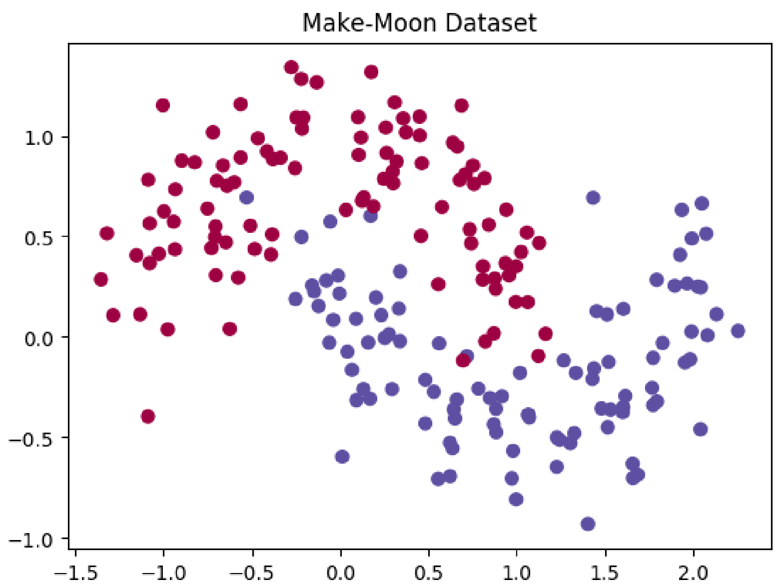


Figure 1: Make Moon Dataset

1b) Derivatives of Activation Functions

Sigmoid:

$$f(x) = \frac{1}{1+e^{-x}} = (1+e^{-x})^{-1}$$

$$\frac{d(1+e^{-x})^{-1}}{dx} = (1+e^{-x})^{-2}(e^{-x})$$

$$= \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}}$$

$$= \frac{1}{1+e^{-x}}\frac{(1+e^{-x})-1}{1+e^{-x}}$$

$$= \frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right)$$

$$\boldsymbol{\frac{d(1+e^{-x})^{-1}}{dx} = f(x)(1-f(x))}$$

Tanh:

$$f(x) = \tanh x = \frac{\sinh x}{\cosh x}$$

$$\frac{d\left(\frac{\sinh x}{\cosh x}\right)}{dx} = \frac{\cosh x\cosh x - \sinh x\sinh x}{\cosh^2 x}$$

$$= \frac{\cosh^2 x - \sinh^2 x}{\cosh^2 x} = \frac{\cosh^2 x}{\cosh^2 x} - \frac{\sinh^2 x}{\cosh^2 x}$$

$$\boldsymbol{\frac{d\left(\frac{\sinh x}{\cosh x}\right)}{dx} = 1 - \tanh^2 x}$$

ReLu:

$$f(x) = max(0, x)$$

$$f(x) = \begin{cases} x, & x > 0 \\ 0, & \text{otherwise} \end{cases}$$

$$f'(x) = \begin{cases} 1, & x > 0 \\ 0, & \text{otherwise} \end{cases}$$

1c) Building the Neural Network

Three Layer Network

$$z^1 = W^1 x + b^1 \tag{1}$$
$$a^1 = actFun(z^1) \tag{2}$$
$$z^2 = W^2 a^1 + b^2 \tag{3}$$
$$a^2 = \hat{y} = softmax(z^2) \tag{4}$$

Mean Cross Entropy Loss of Batch

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{C} y_{n,i} \log \hat{y}_{n,i} \tag{5}$$

1d) Backward Pass - Backpropagation

Gradients: $\frac{\partial L}{\partial W^2}$, $\frac{\partial L}{\partial b^2}$, $\frac{\partial L}{\partial W^1}$, $\frac{\partial L}{\partial b^1}$

$$\frac{\partial L}{\partial W^2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^2} \frac{\partial z^2}{\partial W^2}$$

$$\frac{\partial z^2}{\partial W^2} = \frac{\partial (W^2 a^2 + b^2)}{\partial W^2}$$

$$\boldsymbol{\frac{\partial z^2}{\partial W^2} = a^2}$$

$$\frac{\partial \hat{y}_i}{\partial z_i^2} = \frac{\partial \left(\frac{e^{z_i^2}}{\sum_{j=1}^{C} e^{z_j^2}} \right)}{\partial z_i^2}$$

if $j = i$

$$= \frac{e^{z_i^2} \sum_{j=1}^{C} e^{z_j^2} - e^{z_i^2} e^{z_i^2}}{\left(\sum_{j=1}^{C} e^{z_j^2} \right)^2} = \frac{e^{z_i^2} (\sum_{j=1}^{C} e^{z_j^2} - e^{z_i^2})}{\sum_{j=1}^{C} e^{z_j^2} \sum_{j=1}^{C} e^{z_j^2}}$$

$$\boldsymbol{\frac{\partial \hat{y}_i}{\partial z_i^2} = \hat{y}_i (1 - \hat{y}_i)}$$

if $j \neq i$

$$\frac{\partial \hat{y}_i}{\partial z_j^2} = \frac{0 - e^{z_i^2} e^{z_j^2}}{(\sum_{j=1}^{C} e^{z_j^2})^2}$$

$$\boldsymbol{\frac{\partial \hat{y}_i}{\partial z_j^2} = -\hat{y}_i \hat{y}_j}$$

$$\frac{\partial L}{\partial \hat{y}_i} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{C} \frac{\partial (y_i \log \hat{y}_i)}{\partial \hat{y}_i} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{C} y_i \left(\frac{1}{\hat{y}_i}\right) \frac{\partial \hat{y}_i}{\partial z_i^2}$$

$\forall i$

$$= -\frac{1}{N} \sum_{n=1}^{N} \left[\frac{y_i}{\hat{y}_i} \hat{y}_i \left(1 - \hat{y}_i\right) + \sum_{j \neq i}^{C} \frac{y_j}{\hat{y}_j} (-\hat{y}_j \hat{y}_i) \right]$$

$$= -\frac{1}{N} \sum_{n=1}^{N} \left[y_i (1 - \hat{y}_i) - \sum_{j \neq i}^{C} y_j \hat{y}_i \right]$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left[-y_i + y_i \hat{y}_i + \sum_{j \neq i}^{C} y_j \hat{y}_i \right]$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left[-y_i + \sum_{j=1}^{C} y_j \hat{y}_i \right]$$

$=$ since $\boldsymbol{y_j}$ is one-hot encoded

$$\implies \sum_{j=1}^{C} y_j = 1$$

$$\therefore \boldsymbol{\frac{\partial L}{\partial z_i^2}} = \boldsymbol{\frac{1}{N} \sum_{n=1}^{N} (\hat{y}_i - y_i)}$$

Let $\delta^3 = (\hat{y}_i - y_i)$

$$\frac{\partial L}{\partial W^2} = \frac{1}{N} \sum_{n=1}^{N} \delta^3 a^3$$

$$\boldsymbol{\frac{\partial L}{\partial W^2} = \frac{1}{N} a^{1T} \delta^3}$$

$$\frac{\partial L}{\partial b^2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^2} \frac{\partial z^2}{\partial b^2}$$

$$\frac{\partial z^2}{\partial b^2} = 1$$

$$\boldsymbol{\frac{\partial L}{\partial b^2} = \frac{1}{N} \sum_{n=1}^{C} \delta^3}$$

$$\frac{\partial L}{\partial W^1} = \underbrace{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^2}}_{\frac{1}{N}\sum_{i=1}^{N}\delta^3} \overbrace{\frac{\partial z^2}{\partial a^1}}^{W^2} \underbrace{\frac{\partial a^1}{\partial z^1}}_{f'(z^1)} \overbrace{\frac{\partial z^1}{\partial W^1}}^{x}$$

$$= \frac{1}{N} W^{2T} \delta^3 f'(z^1)$$

$$\textbf{Let } \delta^2 = W^{2T} \delta^3 f'(z^1)$$

$$= \frac{1}{N} \delta^2 x$$

$$\boldsymbol{\frac{\partial L}{\partial W^1} = \frac{1}{N} x^T \delta^2}$$

$$\frac{\partial L}{\partial b^1} = \underbrace{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^2} \frac{\partial z^2}{\partial a^1} \frac{\partial a^1}{\partial z^1}}_{\frac{1}{N}\delta^2} \overbrace{\frac{\partial z^1}{\partial b^1}}^{1}$$

$$\boldsymbol{\frac{\partial L}{\partial b^1} = \frac{1}{N} \delta^2}$$

1e) Training Network

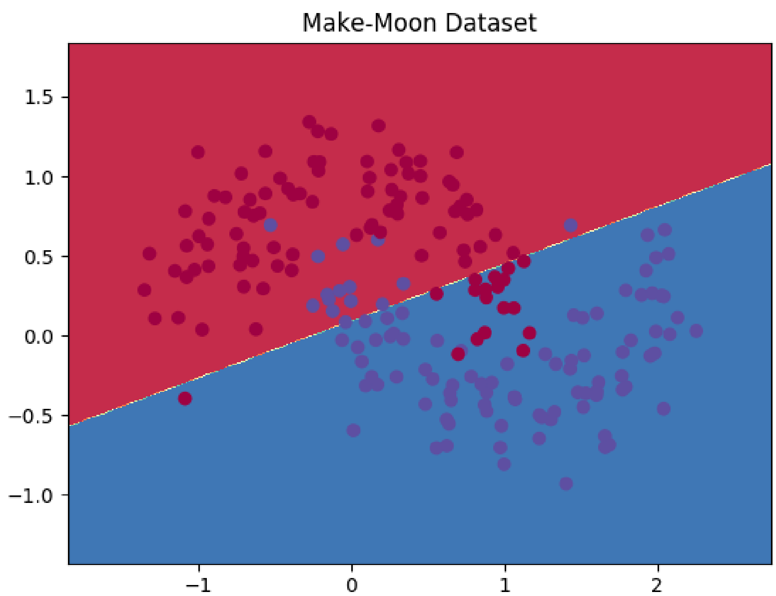


Figure 2: Using Sigmoid Activation Function with 3 Hidden Neurons and stepsize = 0.01

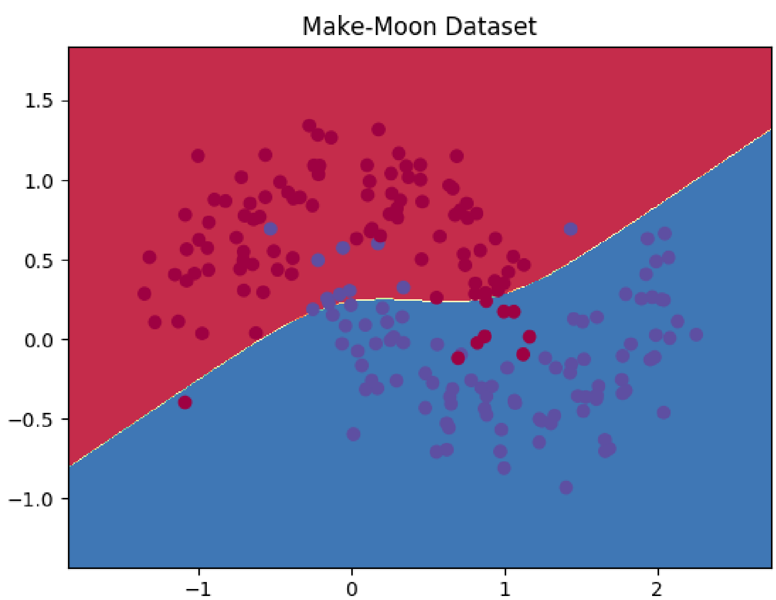


Figure 3: Using Tanh Activation Function with 3 Hidden Neurons and stepsize = 0.01

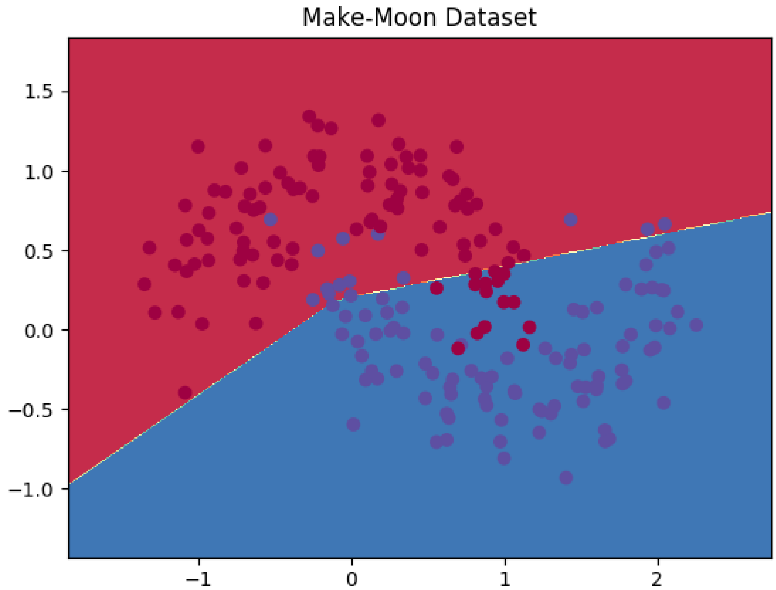


Figure 4: Using ReLu Activation Function with 3 Hidden Neurons and stepsize = 0.01

Training Using more Neurons .

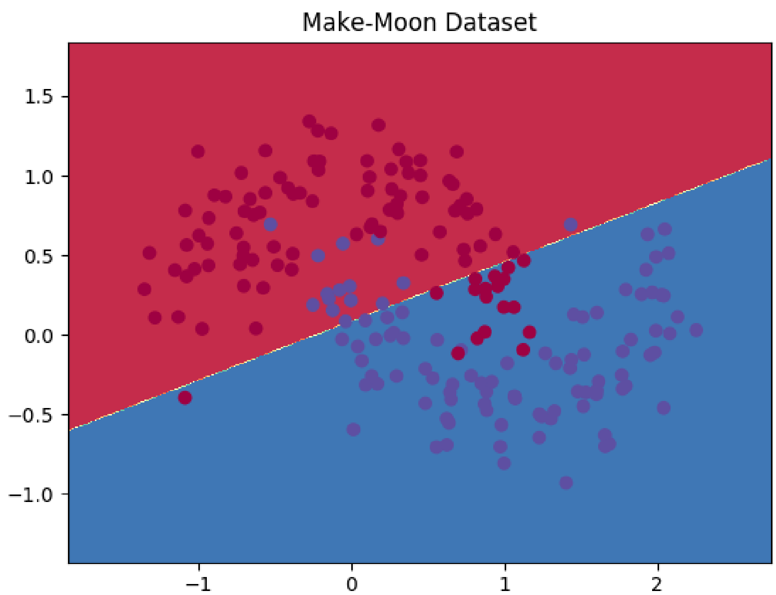


Figure 5: Using Sigmoid Activation Function with 50 Hidden Neurons and stepsize = 0.01

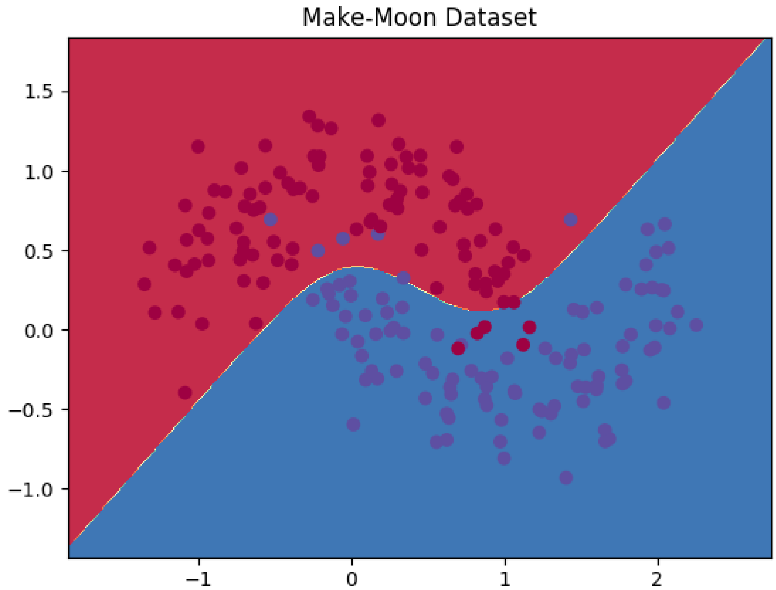


Figure 6: Using Tanh Activation Function with 50 Hidden Neurons and stepsize = 0.01

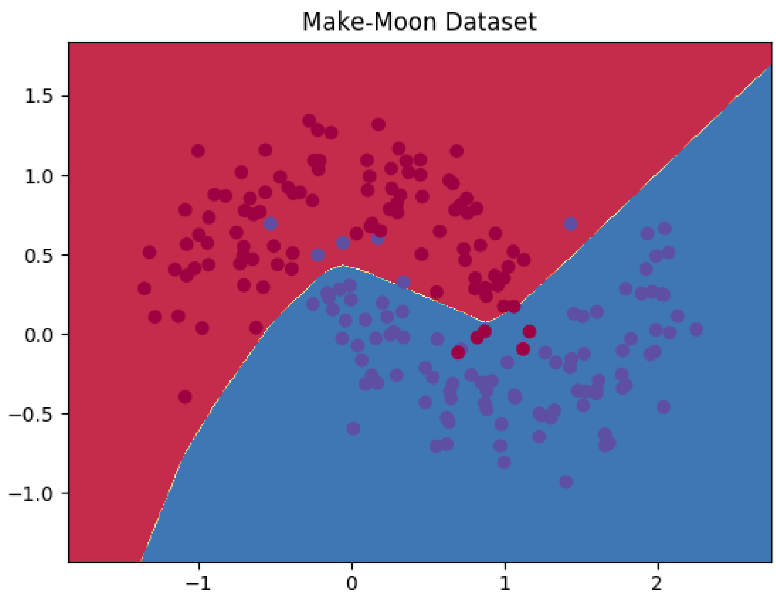


Figure 7: Using ReLu Activation Function with 50 Hidden Neurons and stepsize = 0.01

As you increase the number of neurons in the hidden layer, the decision boundary improves in both cases where the activation function is Tanh and ReLu, where as the Sigmoid case remains constant. In both Tanh and ReLu accuracy increased when the number of hidden was increased.

1f) Training a Deep Network

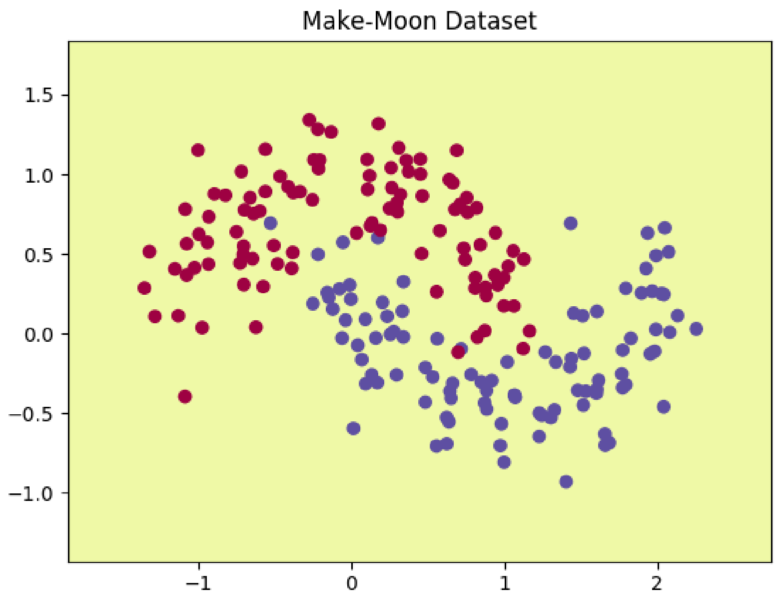


Figure 8: Using Sigmoid Activation Function with 10 Hidden Layers and 20 Hidden Neurons/layer and stepsize = 0.01

The sigmoid function with this architecture performed so poorly that it was not able to draw a decision boundary seperating the dataset

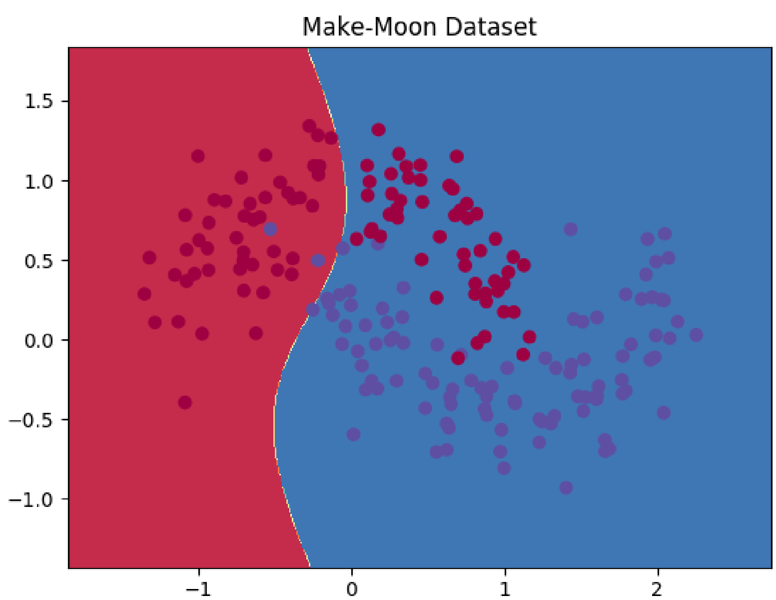


Figure 9: Using Tanh Activation Function with 10 Hidden Layers and 20 Hidden Neurons/layer and stepsize = 0.01

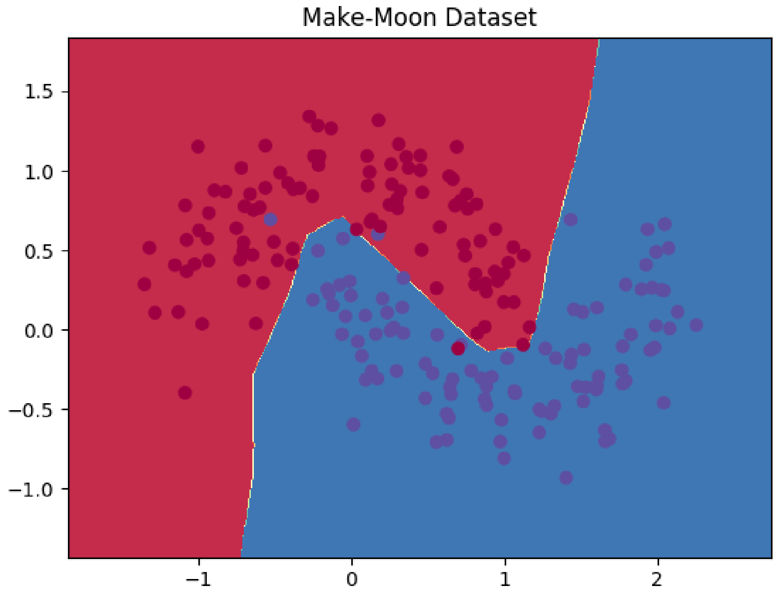


Figure 10: Using ReLu Activation Function with 10 Hidden Layers and 20 Hidden Neurons/layer and stepsize = 0.01

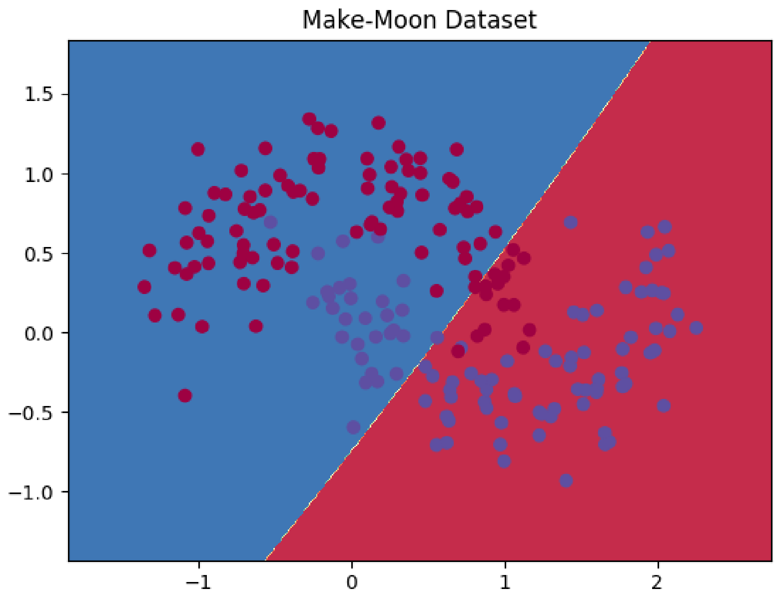


Figure 11: Using Sigmoid Activation Function with 2 Hidden Layers and 10 Hidden Neurons/layer and stepsize = 0.01

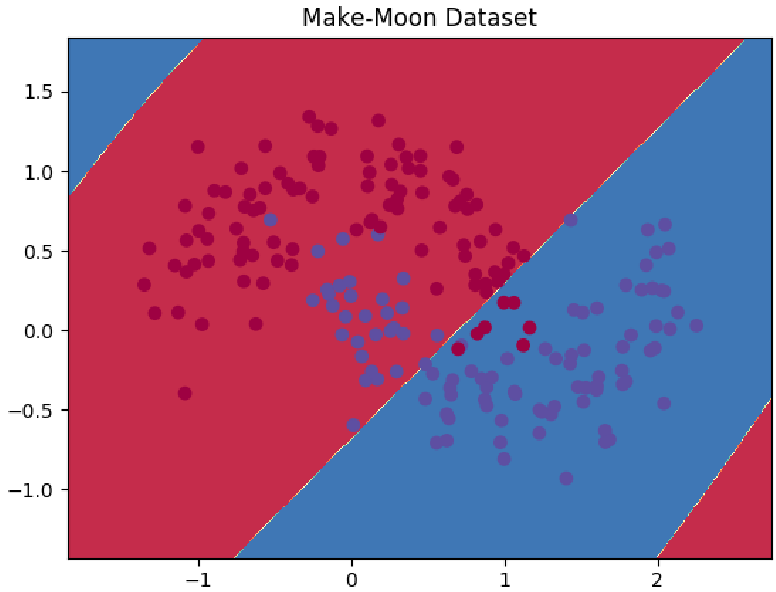


Figure 12: Using Tanh Activation Function with 2 Hidden Layers and 10 Hidden Neurons/layer and stepsize = 0.01

Using a deep Network the ReLu activation function works the best. In both the Sigmoid and Tanh activation cases, having a deeper network produces worse results.

Training with a new dataset. I decided to use the make circle dataset to see how well the network performs where seperation is highly non-linear.

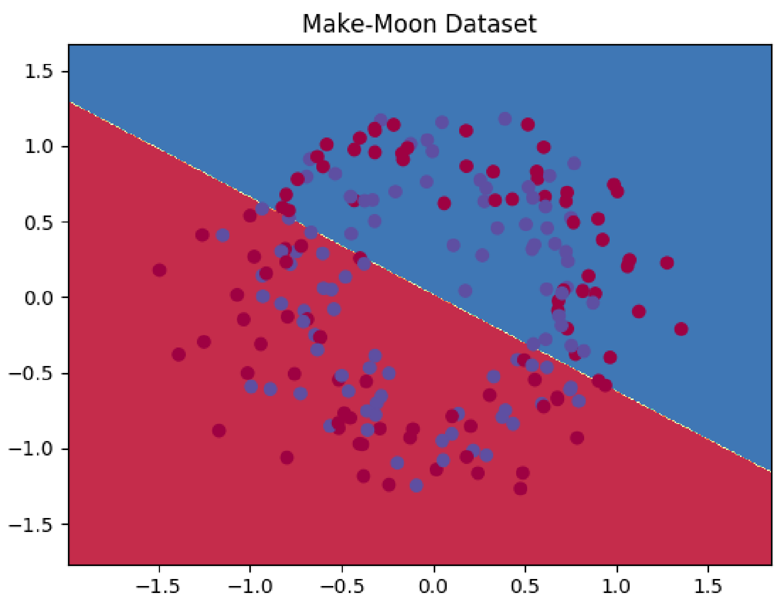


Figure 13: Using Sigmoid Activation Function with 2 Hidden Layers and 10 Hidden Neurons/layer and stepsize = 0.01

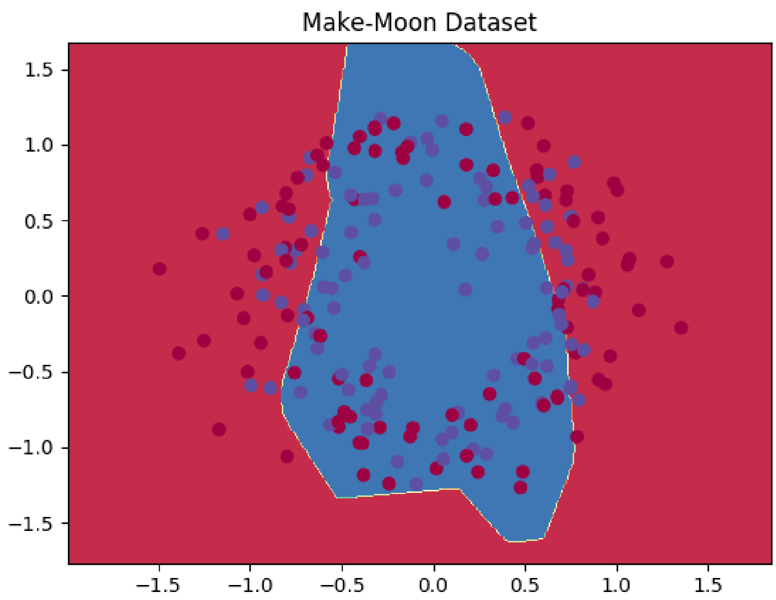


Figure 14: Using ReLu Activation Function with 2 Hidden Layers and 10 Hidden Neurons/layer and stepsize = 0.01

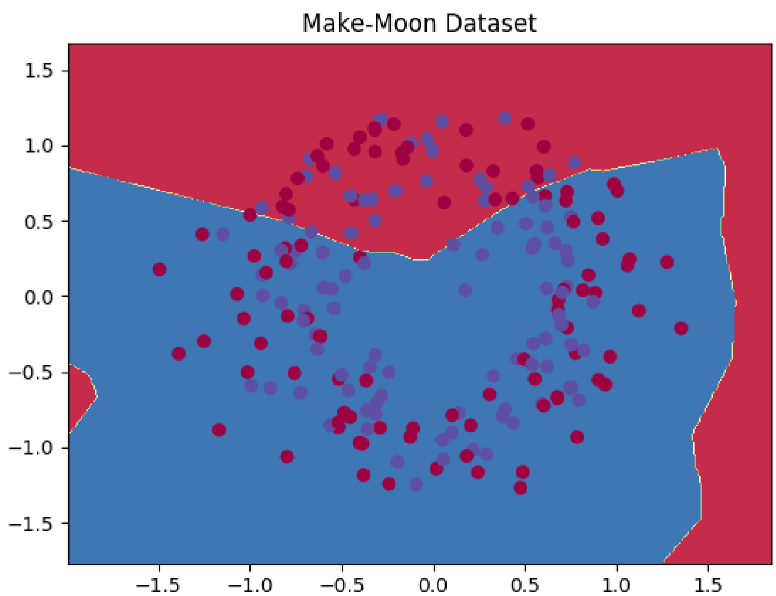


Figure 15: Using ReLu Activation Function with 6 Hidden Layers and 10 Hidden Neurons/layer and stepsize = 0.01

Once again, the ReLu activation functions performs the best. Also as you increase the depth of the network the performance starts the decrease

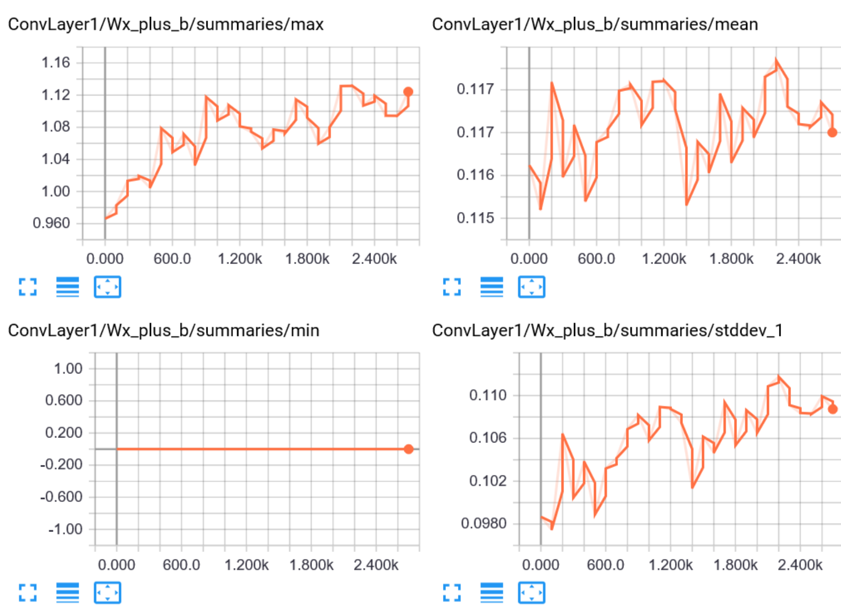
Training a Simple Deep Convolutional Network on MNIST

2.a) Final Test Accuracy

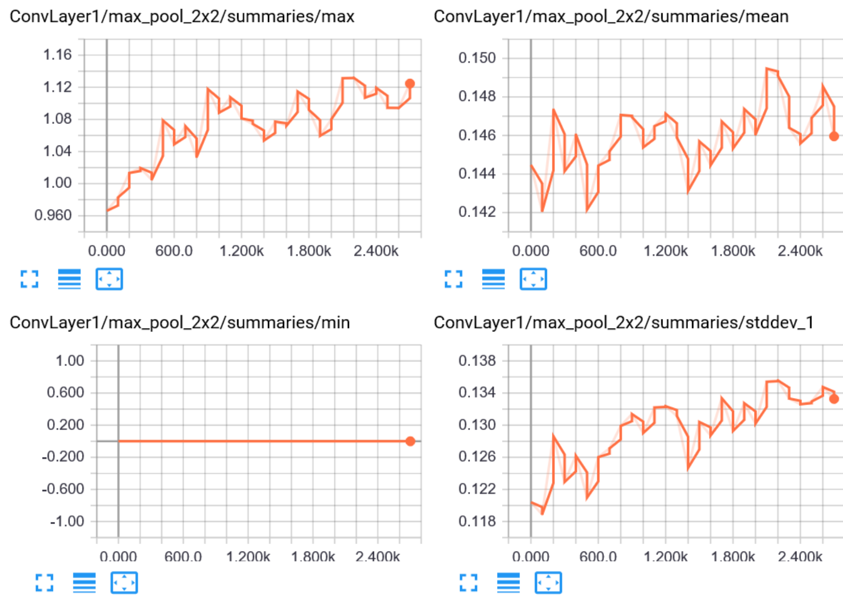
The final Test accuracy for this run is 0.9865

Figure 16: Training Loss

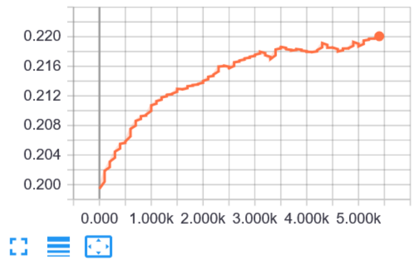
2.b) More on Visualizing Your Training



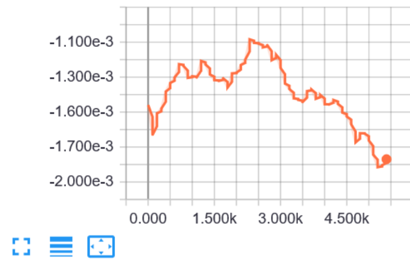

ConvLayer1/biases/summaries/max
0.103
0.102
0.102
0.101
0.000
600.0
1.200k
1.800k
2.400k
ConvLayer1/biases/summaries/mean
0.100
0.0990
0.0980
0.0970
0.0960
0.000
600.0
1.200k
1.800k
2.400k
ConvLayer1/biases/summaries/min
0.0980
0.0960
0.0940
0.0920
0.0900
0.000
600.0
1.200k
1.800k
2.400k
ConvLayer1/biases/summaries/stddev_1
3.000e-3
2.000e-3
1.000e-3
0.00
0.000
600.0
1.200k
1.800k
2.400k


ConvLayer1/max_pool_2x2/summaries/max
1.16
1.12
1.08
1.04
1.00
0.960
0.000
600.0
1.200k
1.800k
2.400k
ConvLayer1/max_pool_2x2/summaries/mean
0.150
0.148
0.146
0.144
0.142
0.000
600.0
1.200k
1.800k
2.400k
ConvLayer1/max_pool_2x2/summaries/min
1.00
0.600
0.200
-0.200
-0.600
-1.00
0.000
600.0
1.200k
1.800k
2.400k
ConvLayer1/max_pool_2x2/summaries/stddev_1
0.138
0.134
0.130
0.126
0.122
0.118
0.000
600.0
1.200k
1.800k
2.400k

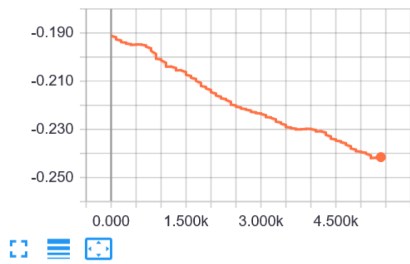
ConvLayer1/weights/summaries/max



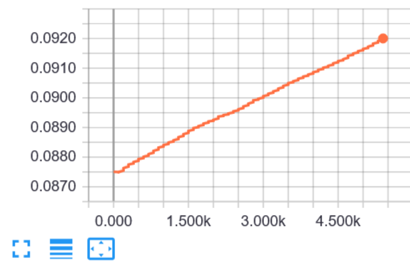
ConvLayer1/weights/summaries/mean

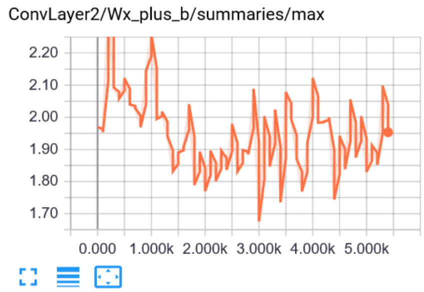


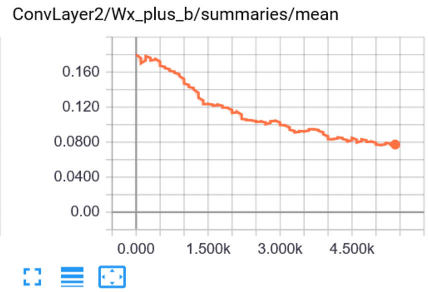
ConvLayer1/weights/summaries/min

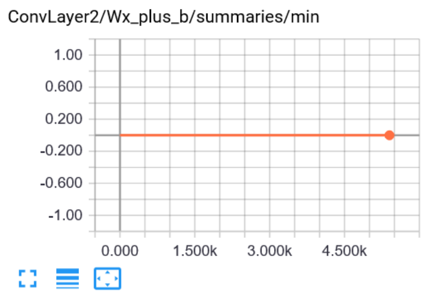


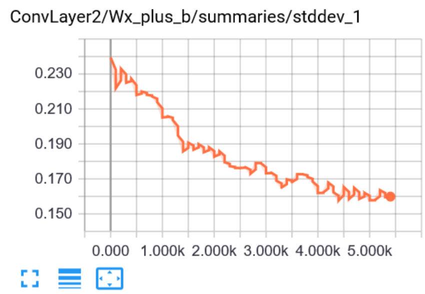
ConvLayer1/weights/summaries/stddev_1

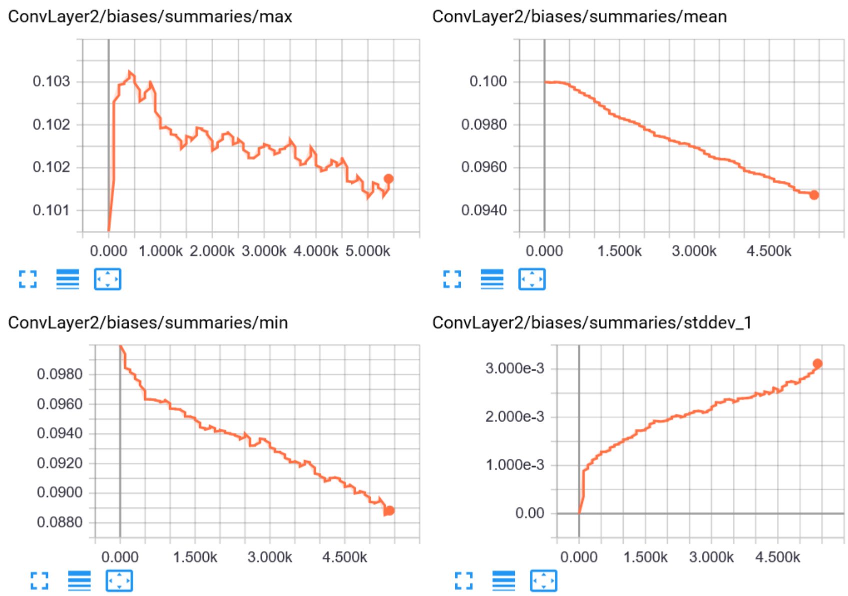


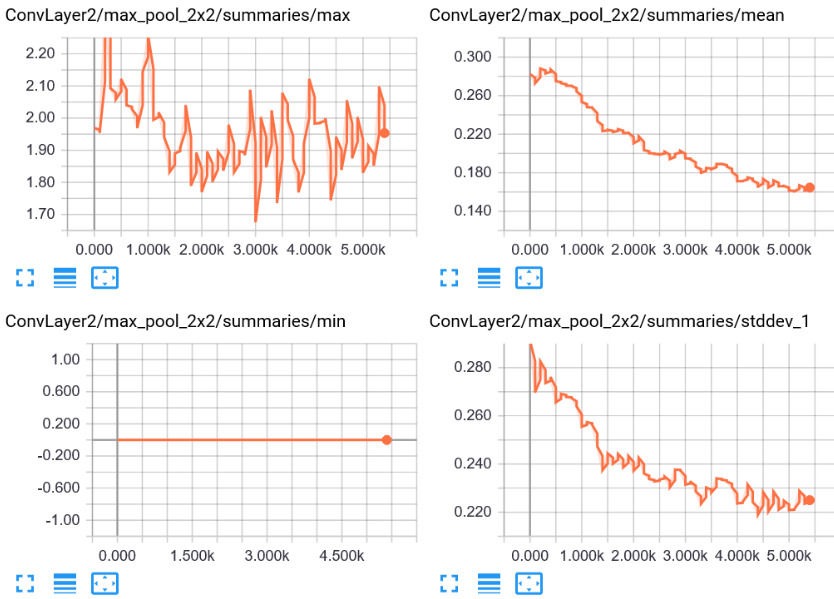

ConvLayer2/Wx_plus_b/summaries/max
2.20
2.10
2.00
1.90
1.80
1.70
0.000 1.000k 2.000k 3.000k 4.000k 5.000k

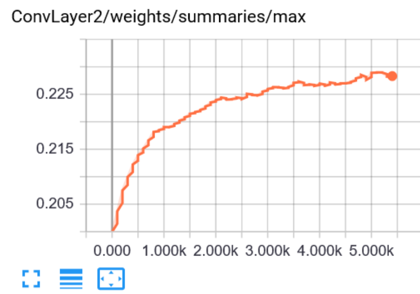

ConvLayer2/Wx_plus_b/summaries/mean
0.160
0.120
0.0800
0.0400
0.00
0.000 1.500k 3.000k 4.500k

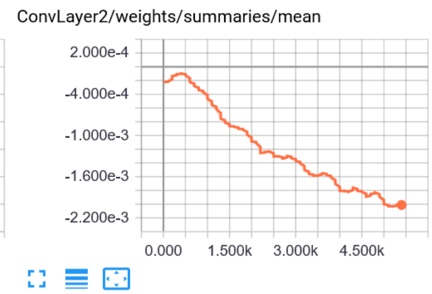

ConvLayer2/Wx_plus_b/summaries/min
1.00
0.600
0.200
-0.200
-0.600
-1.00
0.000 1.500k 3.000k 4.500k

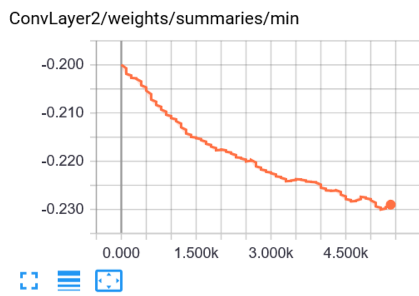

ConvLayer2/Wx_plus_b/summaries/stddev_1
0.230
0.210
0.190
0.170
0.150
0.000 1.000k 2.000k 3.000k 4.000k 5.000k

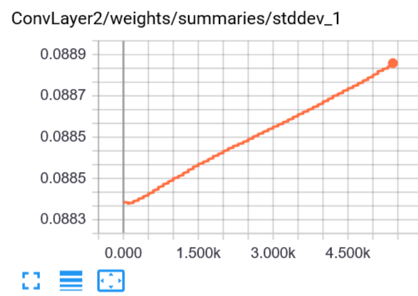

ConvLayer2/biases/summaries/max
0.103
0.102
0.102
0.101
0.000 1.000k 2.000k 3.000k 4.000k 5.000k
ConvLayer2/biases/summaries/mean
0.100
0.0980
0.0960
0.0940
0.000 1.500k 3.000k 4.500k
ConvLayer2/biases/summaries/min
0.0980
0.0960
0.0940
0.0920
0.0900
0.0880
0.000 1.500k 3.000k 4.500k
ConvLayer2/biases/summaries/stddev_1
3.000e-3
2.000e-3
1.000e-3
0.00
0.000 1.500k 3.000k 4.500k

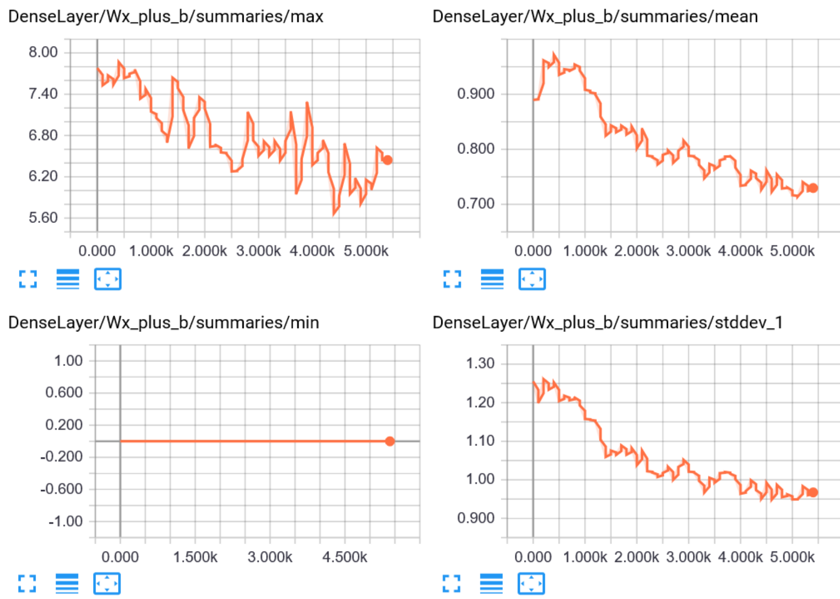

ConvLayer2/max_pool_2x2/summaries/max
2.20
2.10
2.00
1.90
1.80
1.70
0.000 1.000k 2.000k 3.000k 4.000k 5.000k
ConvLayer2/max_pool_2x2/summaries/mean
0.300
0.260
0.220
0.180
0.140
0.000 1.000k 2.000k 3.000k 4.000k 5.000k
ConvLayer2/max_pool_2x2/summaries/min
1.00
0.600
0.200
-0.200
-0.600
-1.00
0.000 1.500k 3.000k 4.500k
ConvLayer2/max_pool_2x2/summaries/stddev_1
0.280
0.260
0.240
0.220
0.000 1.000k 2.000k 3.000k 4.000k 5.000k

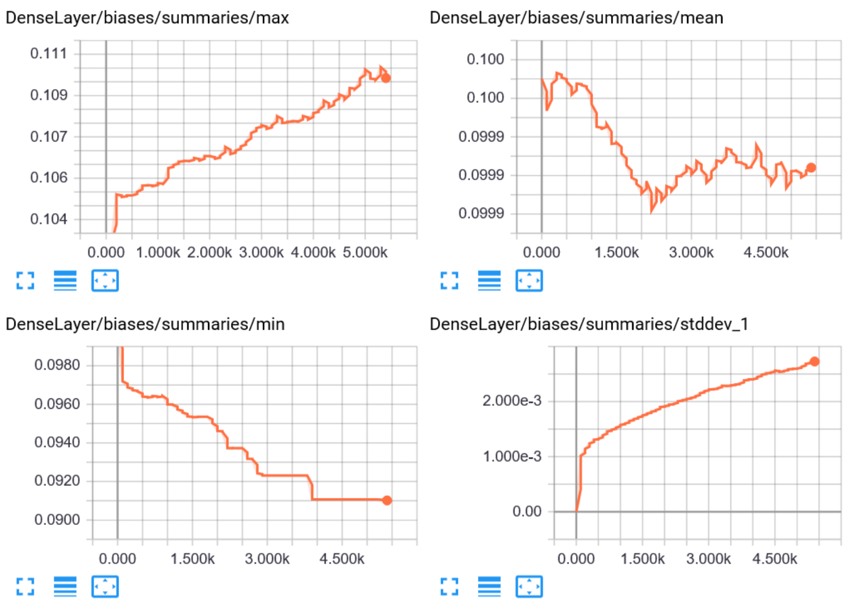

ConvLayer2/weights/summaries/max
0.225
0.215
0.205
0.000 1.000k 2.000k 3.000k 4.000k 5.000k

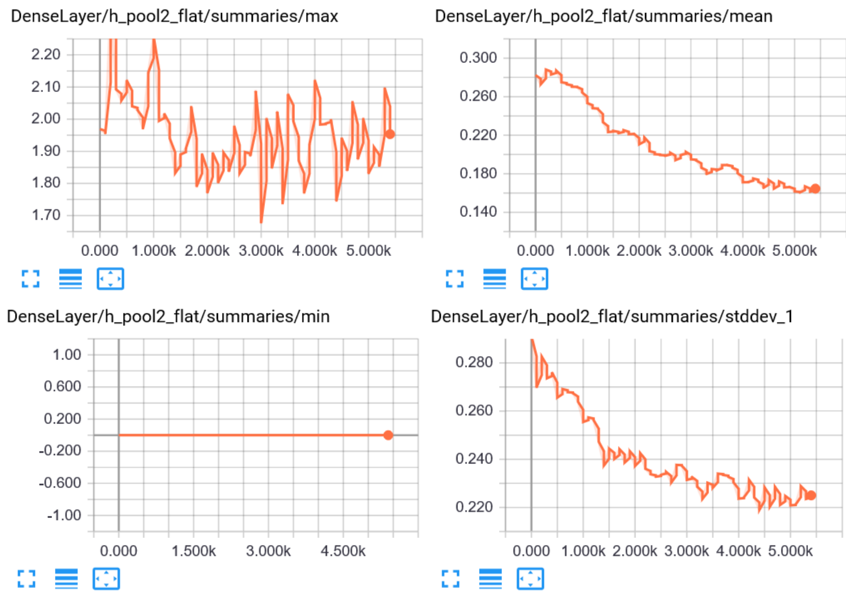

ConvLayer2/weights/summaries/mean
2.000e-4
-4.000e-4
-1.000e-3
-1.600e-3
-2.200e-3
0.000 1.500k 3.000k 4.500k

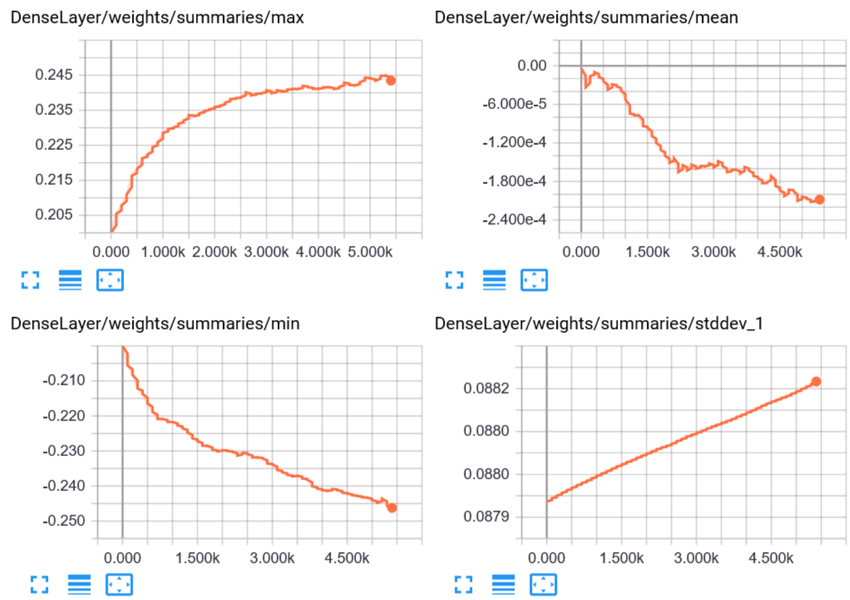

ConvLayer2/weights/summaries/min
-0.200
-0.210
-0.220
-0.230
0.000 1.500k 3.000k 4.500k


ConvLayer2/weights/summaries/stddev_1
0.0889
0.0887
0.0885
0.0885
0.0883
0.000 1.500k 3.000k 4.500k


DenseLayer/Wx_plus_b/summaries/max
8.00
7.40
6.80
6.20
5.60
0.000 1.000k 2.000k 3.000k 4.000k 5.000k
DenseLayer/Wx_plus_b/summaries/mean
0.900
0.800
0.700
0.000 1.000k 2.000k 3.000k 4.000k 5.000k
DenseLayer/Wx_plus_b/summaries/min
1.00
0.600
0.200
-0.200
-0.600
-1.00
0.000 1.500k 3.000k 4.500k
DenseLayer/Wx_plus_b/summaries/stddev_1
1.30
1.20
1.10
1.00
0.900
0.000 1.000k 2.000k 3.000k 4.000k 5.000k

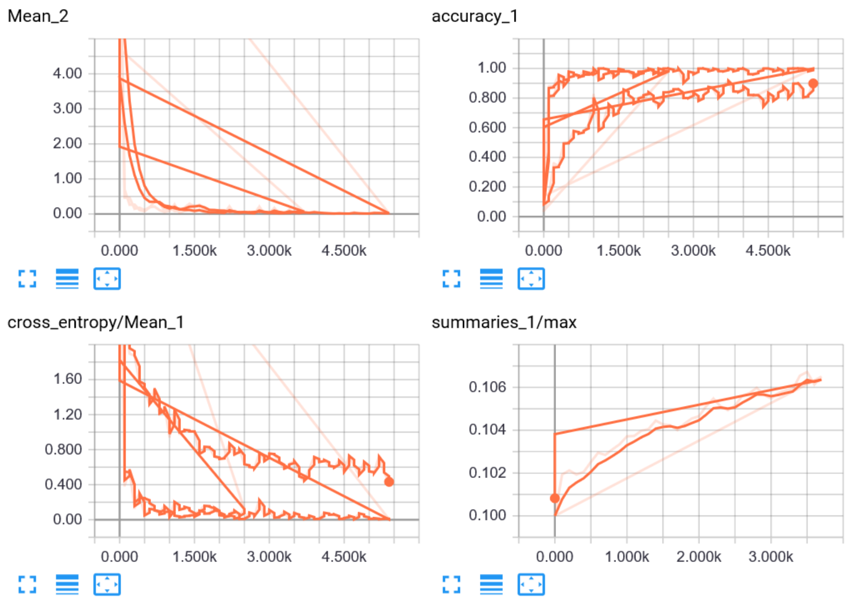

DenseLayer/biases/summaries/max
0.111
0.109
0.107
0.106
0.104
0.000 1.000k 2.000k 3.000k 4.000k 5.000k
DenseLayer/biases/summaries/mean
0.100
0.100
0.0999
0.0999
0.0999
0.000 1.500k 3.000k 4.500k
DenseLayer/biases/summaries/min
0.0980
0.0960
0.0940
0.0920
0.0900
0.000 1.500k 3.000k 4.500k
DenseLayer/biases/summaries/stddev_1
2.000e-3
1.000e-3
0.00
0.000 1.500k 3.000k 4.500k


DenseLayer/h_pool2_flat/summaries/max
2.20
2.10
2.00
1.90
1.80
1.70
0.000 1.000k 2.000k 3.000k 4.000k 5.000k
DenseLayer/h_pool2_flat/summaries/mean
0.300
0.260
0.220
0.180
0.140
0.000 1.000k 2.000k 3.000k 4.000k 5.000k
DenseLayer/h_pool2_flat/summaries/min
1.00
0.600
0.200
-0.200
-0.600
-1.00
0.000 1.500k 3.000k 4.500k
DenseLayer/h_pool2_flat/summaries/stddev_1
0.280
0.260
0.240
0.220
0.000 1.000k 2.000k 3.000k 4.000k 5.000k

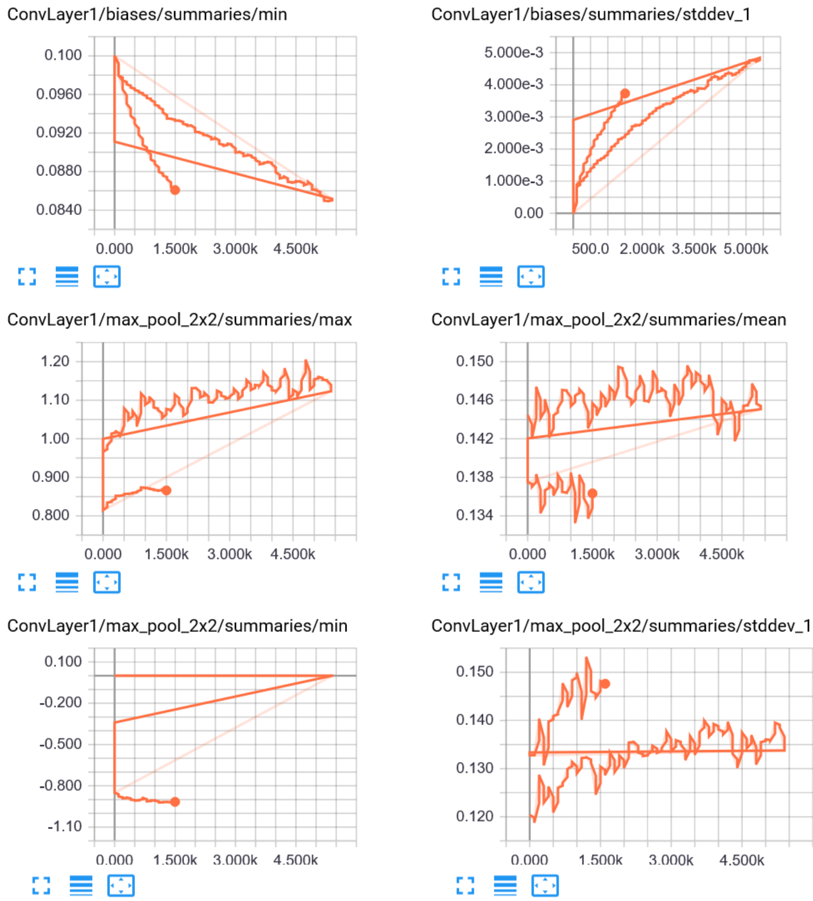


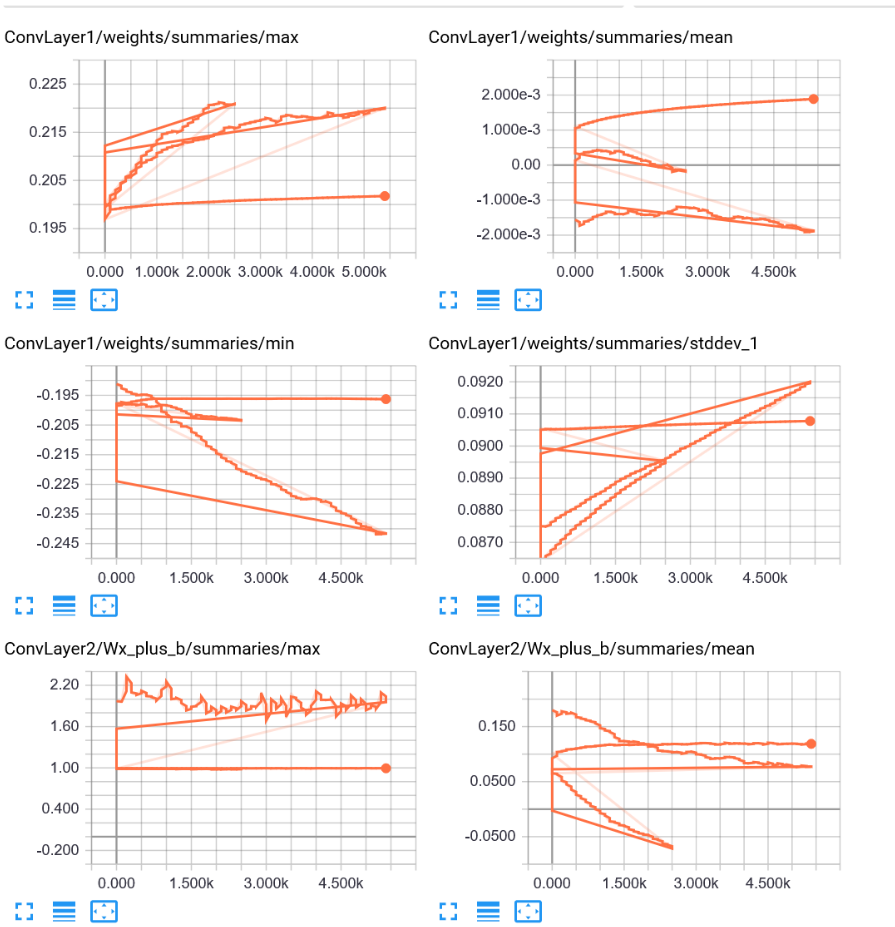
2.c) Time for More Fun!!!

Training with Tanh activation function and momentum optimizer. A final accuracy of 0.9125

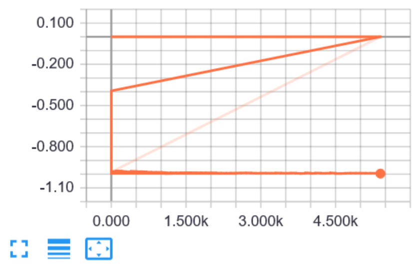

Mean_2
4.00
3.00
2.00
1.00
0.00
0.000
1.500k
3.000k
4.500k
accuracy_1
1.00
0.800
0.600
0.400
0.200
0.00
0.000
1.500k
3.000k
4.500k
cross_entropy/Mean_1
1.60
1.20
0.800
0.400
0.00
0.000
1.500k
3.000k
4.500k
summaries_1/max
0.106
0.104
0.102
0.100
0.000
1.000k
2.000k
3.000k


ConvLayer1/Wx_plus_b/summaries/max
1.20
1.10
1.00
0.900
0.800
0.000
1.500k
3.000k
4.500k
ConvLayer1/Wx_plus_b/summaries/mean
0.118
0.112
0.106
0.100
0.0940
0.000
1.500k
3.000k
4.500k
ConvLayer1/Wx_plus_b/summaries/min
0.100
-0.200
-0.500
-0.800
-1.10
0.000
1.500k
3.000k
4.500k
ConvLayer1/Wx_plus_b/summaries/stddev_1
0.160
0.140
0.120
0.100
0.000
1.500k
3.000k
4.500k
ConvLayer1/biases/summaries/max
0.105
0.104
0.103
0.102
0.102
0.101
0.000
1.500k
3.000k
4.500k
ConvLayer1/biases/summaries/mean
0.0990
0.0970
0.0950
0.0930
0.000
1.500k
3.000k
4.500k

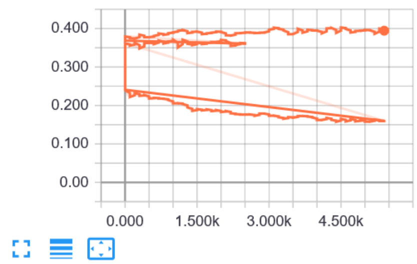

ConvLayer1/biases/summaries/min
0.100
0.0960
0.0920
0.0880
0.0840
0.000 1.500k 3.000k 4.500k
ConvLayer1/biases/summaries/stddev_1
5.000e-3
4.000e-3
3.000e-3
2.000e-3
1.000e-3
0.00
500.0 2.000k 3.500k 5.000k
ConvLayer1/max_pool_2x2/summaries/max
1.20
1.10
1.00
0.900
0.800
0.000 1.500k 3.000k 4.500k
ConvLayer1/max_pool_2x2/summaries/mean
0.150
0.146
0.142
0.138
0.134
0.000 1.500k 3.000k 4.500k
ConvLayer1/max_pool_2x2/summaries/min
0.100
-0.200
-0.500
-0.800
-1.10
0.000 1.500k 3.000k 4.500k
ConvLayer1/max_pool_2x2/summaries/stddev_1
0.150
0.140
0.130
0.120
0.000 1.500k 3.000k 4.500k


ConvLayer1/weights/summaries/max
0.225
0.215
0.205
0.195
0.000 1.000k 2.000k 3.000k 4.000k 5.000k
ConvLayer1/weights/summaries/mean
2.000e-3
1.000e-3
0.00
-1.000e-3
-2.000e-3
0.000 1.500k 3.000k 4.500k
ConvLayer1/weights/summaries/min
-0.195
-0.205
-0.215
-0.225
-0.235
-0.245
0.000 1.500k 3.000k 4.500k
ConvLayer1/weights/summaries/stddev_1
0.0920
0.0910
0.0900
0.0890
0.0880
0.0870
0.000 1.500k 3.000k 4.500k
ConvLayer2/Wx_plus_b/summaries/max
2.20
1.60
1.00
0.400
-0.200
0.000 1.500k 3.000k 4.500k
ConvLayer2/Wx_plus_b/summaries/mean
0.150
0.0500
-0.0500
0.000 1.500k 3.000k 4.500k

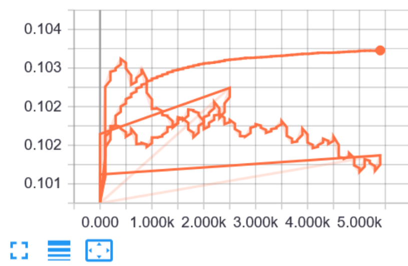
ConvLayer2/Wx_plus_b/summaries/min



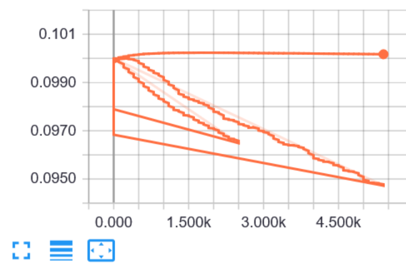
ConvLayer2/Wx_plus_b/summaries/stddev_1



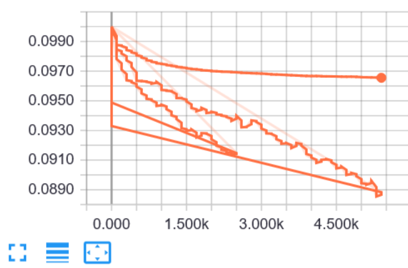
ConvLayer2/biases/summaries/max



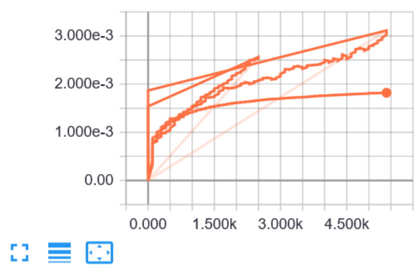
ConvLayer2/biases/summaries/mean



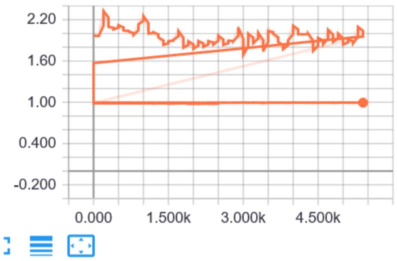
ConvLayer2/biases/summaries/min



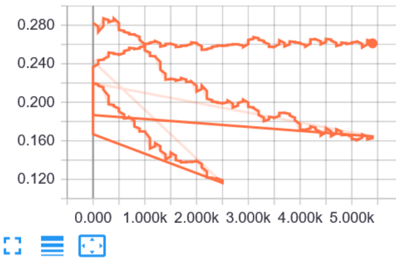
ConvLayer2/biases/summaries/stddev_1



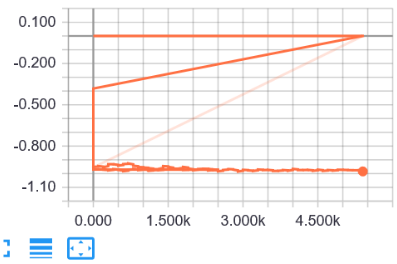
ConvLayer2/max_pool_2x2/summaries/max



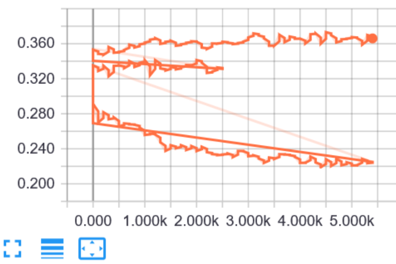
ConvLayer2/max_pool_2x2/summaries/mean



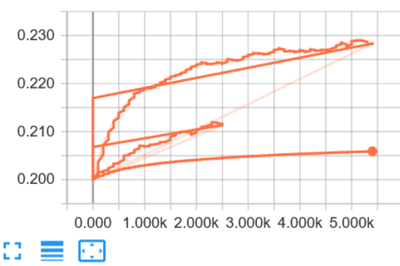
ConvLayer2/max_pool_2x2/summaries/min



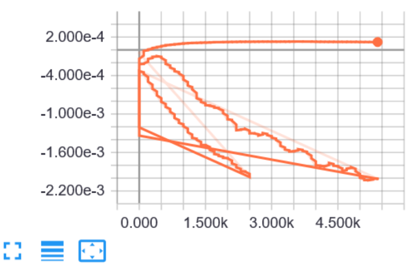
ConvLayer2/max_pool_2x2/summaries/stddev_1



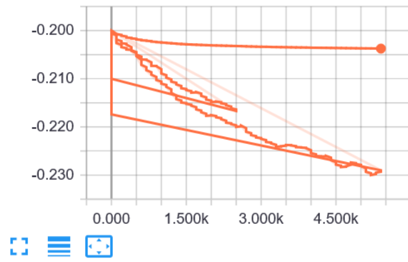
ConvLayer2/weights/summaries/max



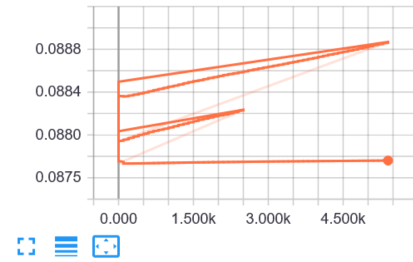
ConvLayer2/weights/summaries/mean



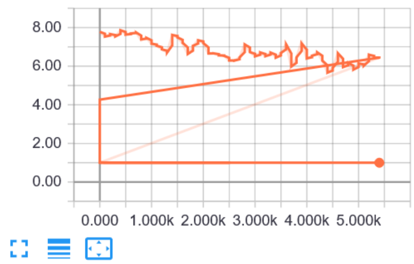
ConvLayer2/weights/summaries/min



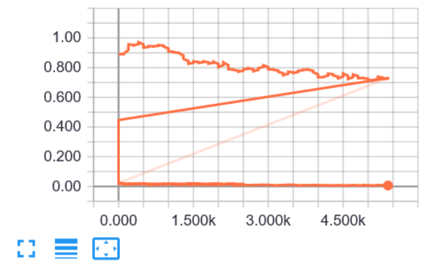
ConvLayer2/weights/summaries/stddev_1



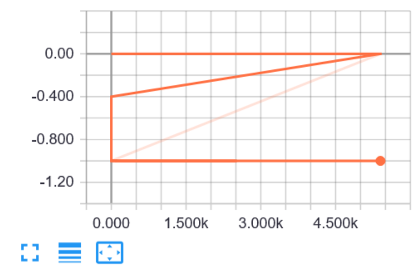
DenseLayer/Wx_plus_b/summaries/max



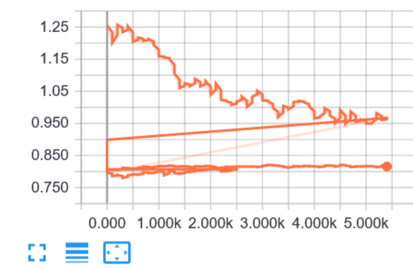
DenseLayer/Wx_plus_b/summaries/mean



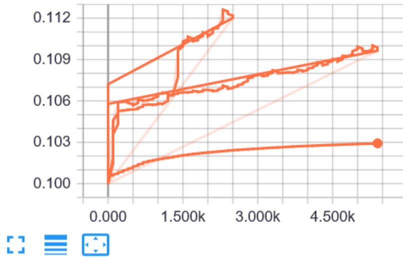
DenseLayer/Wx_plus_b/summaries/min



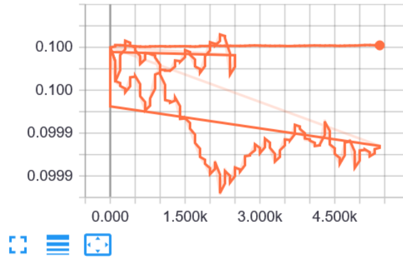
DenseLayer/Wx_plus_b/summaries/stddev_1



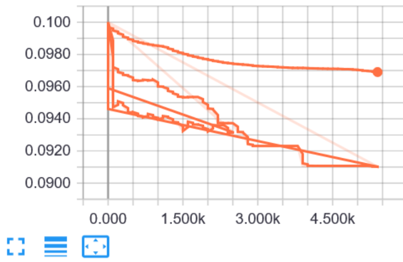
DenseLayer/biases/summaries/max



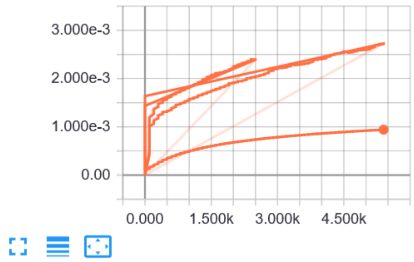
DenseLayer/biases/summaries/mean



DenseLayer/biases/summaries/min



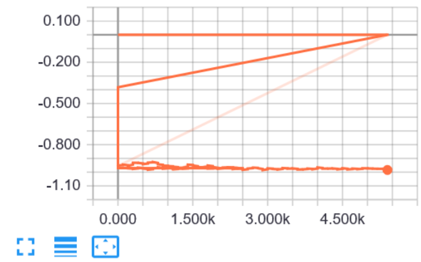
DenseLayer/biases/summaries/stddev_1



DenseLayer/h_pool2_flat/summaries/max

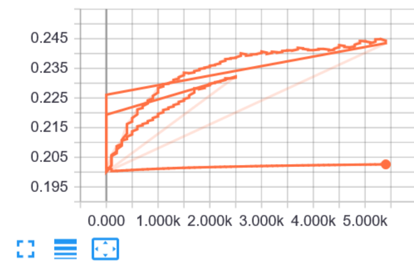
DenseLayer/h_pool2_flat/summaries/mean

DenseLayer/h_pool2_flat/summaries/min

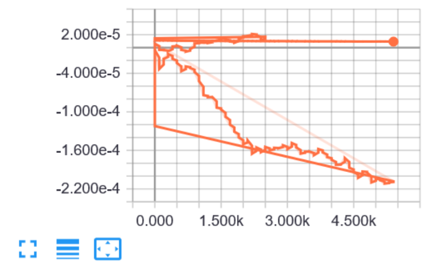


DenseLayer/h_pool2_flat/summaries/stddev_1

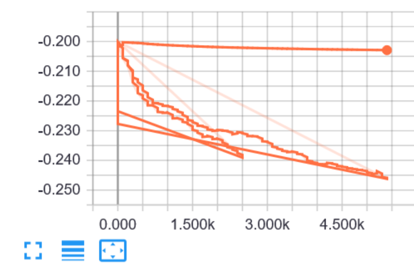
DenseLayer/weights/summaries/max



DenseLayer/weights/summaries/mean



DenseLayer/weights/summaries/min



DenseLayer/weights/summaries/stddev_1